applied sciences

MDPI

Article

Single Trace Analysis on Constant Time CDT Sampler and Its Countermeasure

Suhri Kim and Seokhie Hong *

Center for Information Security Technologies (CIST), Korea University, Seoul 02841, Korea; suhrikim@gmail.com
* Correspondence: shhong@korea.ac.kr

Received: 17 September 2018; Accepted: 29 September 2018; Published: 3 October 2018

Abstract: The Gaussian sampler is an integral part in lattice-based cryptography as it has a direct connection to security and efficiency. Although it is theoretically secure to use the Gaussian sampler, the security of its implementation is an open issue. Therefore, researchers have started to investigate the security of the Gaussian sampler against side-channel attacks. Since the performance of the Gaussian sampler directly affects the performance of the overall cryptosystem, countermeasures considering only timing attacks are applied in the literature. In this paper, we propose the first single trace power analysis attack on a constant-time cumulative distribution table (CDT) sampler used in lattice-based cryptosystems. From our analysis, we were able to recover every sampled value in the key generation stage, so that the secret key is recovered by the Gaussian elimination. By applying our attack to the candidates submitted to the National Institute of Standards and Technology (NIST), we were able to recover over 99% of the secret keys. Additionally, we propose a countermeasure based on a look-up table. To validate the efficiency of our countermeasure, we implemented it in Lizard and measure its performance. We demonstrated that the proposed countermeasure does not degrade the performance.

Keywords: post-quantum cryptography; lattice-based cryptography; Gaussian sampling; CDT sampling; side-channel attack; single trace analysis

1. Introduction

The security of currently used public-key cryptosystems is based on the hardness of mathematical problems such as integer factorization or a discrete logarithm problem over a finite field. For example, RSA, proposed in 1977, is based on the hardness of factoring large integers. Elliptic curve cryptography (ECC), proposed in 1985, is based on the hardness of solving discrete logarithm problem on an elliptic curve (ECDLP). Since the currently best-known algorithms for solving integer factorization and ECDLP take sub-exponential time and exponential time, respectively, RSA and ECC are believed to be secure when using large enough parameters. However, due to the seminal work of Peter Shor in 1994, these problems can be solved in polynomial time when a quantum computer is built running the Shor algorithm [1]. For this reason, quantum-resistant cryptographic schemes have become an active area of research due to the increase in concerns over the security of current public-key cryptosystems.

Post-quantum cryptography (PQC) refers to cryptographic algorithms executed on a classical computer that is expected to be secure against adversaries with quantum computers. PQC is considered to be secure against quantum computers since there is no known quantum algorithm that can solve security base problems more efficiently than classical algorithms. Five categories are mainly considered in post-quantum cryptography: multivariate-based cryptography, code-based cryptography, lattice-based cryptography, hash-based digital signature, and isogeny-based cryptography. Multivariate-based cryptography is based on the hardness of solving a system of multivariate equations, while code-based cryptography is based on the hardness of decoding

general linear code. Lattice-based cryptography is based on the hardness of solving lattice problems and hash-based digital signature is based on the security of cryptographic hash functions. Lastly, isogeny-based cryptography is based on the hardness of finding isogeny between two given elliptic curves. Each cryptosystem in its category has both pros and cons, and no cryptosystem is known to be better than the others. After the National Institute of Standards and Technology (NIST) announced a standardization project for PQC in 2016, its first submission ended in November 2017. Among the 69 submissions, 26 were lattice-based cryptosystems that ranked the highest. Lattice-based cryptography was first proposed by Ajtai and Dwork in 1997 [2]. Later on, due to the groundbreaking work of Regev in 2005 and 2010, lattice-based cryptography became one of the most promising candidates in PQC because of its efficiency [3,4]. By using the learning with errors (LWE) problem over an ideal lattice as a security base, the key size and ciphertext size have decreased significantly compared with Ajtai and Dwork's proposal. Therefore, most cryptographic schemes in lattice-based cryptography nowadays are based on the LWE problem.

Intuitively, the LWE problem aims to find a solution s given a sequence of "approximate" random linear equations on s. That is, given a set of linear equations on s—e.g., $As = B$, where A and B are known, we insert an error e to make the problem significantly more difficult. In this regard, the attacker now needs to solve $As + e = B$, where e is unknown to him. Components of this error vectors are selected from a Gaussian distribution. If an attacker knows all the values of error vectors, the system can be easily broken. In this regard, the closer the sampler is to the actual Gaussian distribution, the more secure the cryptosystem is. At the same time, since the Gaussian sampler is one of the main modules in lattice-based cryptography, it must be efficient to avoid performance degradation. Therefore, previous works have been conducted on efficiency and accuracy (i.e., closeness to the ideal Gaussian distribution) trade-off for Gaussian sampling algorithms [5–8]. Recently, security against side-channel attacks on a Gaussian sampler has been recognized as an important problem.

Just as side-channel analysis has been thoroughly performed on classical cryptosystems such as RSA or ECC, we also need to examine the case in lattice-based cryptography. A side-channel attack proposed by Kocher et al. in 1996 is an attack based on exploiting information gained from the physical implementation of a cryptographic system [9]. In order to substitute RSA or ECC with post-quantum cryptography, resistance against side-channel analysis is required. Hence, researchers are examining possible side-channel attacks on lattice-based cryptography. Particularly, side-channel attacks on Gaussian sampling have been proposed [7,10,11]. Since the Gaussian sampler uses different random values each time of execution, side-channel attacks that exploit multiple iterations of an algorithm such as differential power analysis (DPA) cannot be performed. Moreover, since efficiency is as important as security when implementing the Gaussian sampler, countermeasures that significantly degrade the performance cannot be used. In this regard, current implementation on Gaussian samplers mainly considers timing attacks so that constant-time algorithms on Gaussian sampling have been proposed [7,12–14]. However, single-trace attacks targeting the Gaussian sampler have not been analyzed so far.

The goal of this work is to examine the vulnerability of the current implementation of CDT sampler and to provide its countermeasure. Due to its efficiency, the CDT sampler is one of the most widely used algorithms. As stated above, a constant-time CDT sampler is implemented to withstand timing side-channel attacks. The following list details the main contributions of this work.

- We performed the first single trace analysis (STA) on a constant-time CDT sampler. We theoretically and experimentally analyzed that existing a constant-time CDT algorithm is vulnerable to STA. Our attack is simple as it does not have any restriction on the attack environment. We exploit the gap of Hamming weight between the positive and negative value to recover the sampled value through a single power consumption trace. When the errors are known, the attacker can reveal the secret key by solving $As = B - e$. Therefore, every lattice-based cryptosystem including key exchange is vulnerable to STA. Details of our attack are presented in Section 3.

- We proposed an efficient countermeasure based on look-up table schemes. Especially for the Gaussian sampling algorithm used in lattice-based cryptography, the countermeasures must be as efficient as possible since every entry of an error matrix used in the cryptosystem requires executions of the sampler individually. In this regard, at least 2000 samplings are required for single executions of an algorithm proposed nowadays. Our countermeasure is efficient since it is faster than the constant-time CDT sampler, with the use of ROM. However, the additional memory usage is minimal—only about 256 to 2048 bytes, depending on the choice of parameter. To validate the efficiency of our countermeasure, we implemented it in FrodoKEM and Lizard and compared its speed.
- We present an implementation result to validate the efficiency of our proposed countermeasure. We implemented our countermeasure in Lizard, one of the submitted candidates in NIST standardization projects [15]. From our result, we demonstrated that our countermeasure does not degrade the performance of the constant-time CDT sampler. The result of our implementation is given in Section 4.

This paper is organized as follows: Section 2 recalls the definition of a lattice and describes lattice-based cryptography and a CDT sampler. In Section 3, our attack on constant-time CDT sampler is described. We propose our countermeasure in Section 4 and conclude in Section 5.

2. Preliminaries

In this section, we briefly introduce lattice and lattice-based cryptography. Next, we describe the CDT sampling method. Although there are several sampling methods in the literature, we focused on the CDT sampler since it is most widely used due to its efficiency. In the last subsection, we describe timing a side-channel attack performed in the CDT sampler and its countermeasure.

2.1. Lattice-Based Cryptography

Intuitively, a lattice can be defined as a discrete subgroup of $\mathbb{R}^n$ or a set of integer combinations of basis. Below is the formal definition of a lattice.

Definition 1. *Let $B = \{b_1, ..., b_d\} \subset \mathbb{Z}^n$ be a set of d independent vectors. Then, the set*

$$L(\{b_1, ..., b_d\}) = \left\{ \sum_{i=1}^{d} x_i \cdot b_i | x_i \in \mathbb{Z} \right\} \quad (1)$$

is a lattice and B is called a basis of L.

Given two bases B and B', B and B' generate the same lattice if and only if there exists a unimodular matrix U such that $B = UB'$. It is known that there are infinitely many bases for a given lattice L, and some bases are considered as "good" and many are considered as "bad". Using a "good" basis, one can solve lattice problems more efficiently. Below are definitions of some of the computational problems on lattices.

Definition 2. *(Shortest Vector Problem, SVP) Given an arbitrary basis B of some lattice $L = L(B)$, find a shortest nonzero lattice vector.*

Definition 3. *(Closest Vector Problem, CVP) Given an arbitrary basis B of some lattice $L = L(B)$, and a target vector $t \in \mathbb{R}^n$, find some $v \in L$ such that $||t - v|| \leq \gamma \cdot dist(t, L)$, where $dist(L, t) = min_{x \in L}||x - t||$ and $\gamma \geq 1$ is an approximation factor.*

The above problems are known to be NP-hard, and several cryptosystems based on these problems have been proposed. Since there is no quantum algorithm to solve these problems efficiently,

such lattice-based cryptosystems are secure even against quantum computers. The first public-key cryptosystem based on lattice was proposed by Ajtai and Dwork in 1997 [2]. Although their system was theoretically secure, it was inefficient due to large public-key, private key, and ciphertext sizes. Meanwhile, NTRU was proposed in 1996 by Hoffstein, Pipher, and Silverman [16]. Their system was based on the closest vector problem and has remained secure until now. NTRU is famous for faster operation compared with RSA at equivalent cryptographic strength.

In 2005, Regev proposed that the LWE problem is as hard to solve as several worst-case lattice problems [3]. He showed a quantum reduction from worst-case lattice problems such as SVP to LWE. This LWE problem can be considered as an extension of the "learning parity with noise" problem to higher moduli. Regev further proposed a public-key cryptosystem based on this problem, which is significantly more efficient than the previous lattice-based cryptosystems [3]. The public-key is of size $\tilde{O}(n^2)$ and encrypting a message increases its size by a factor of $\tilde{O}(n)$, where previous cryptosystems were $\tilde{O}(n^4)$ and $\tilde{O}(n^2)$, respectively. Before describing the LWE problem, we first define the LWE distribution.

Definition 4. *(LWE distribution) For a vector $s \in \mathbb{Z}_q^n$, called the secret, the LWE distribution $A_{s,\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is sampled by choosing $a \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \leftarrow \chi$ and generating $(a, b =< s, a > +e \mod q)$.*

In the above definition, χ is an error distribution over $\mathbb{Z}$, usually taken to be a discrete Gaussian distribution. There are two versions of the LWE problem: the search-LWE problem that aims to find the secret vector given LWE samples and the decision-LWE problem that aims to distinguish between LWE samples and uniformly random samples. Below are definitions of search-LWE and decision-LWE problems [17]. In 2010, Lyubashevsky, Peikert, and Regev proposed a ring-base analog of LWE problems, known as the ring LWE problem (RLWE) [4]. Due to its structure and the usage of number theoretic transform (NTT) as an underlying computation, RLWE further improved lattice-based cryptography.

Definition 5. *(Search-LWE problem) Given m independent samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ drawn from $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ (fixed for all samples), find s.*

Definition 6. *(Decision-LWE problem) Given m independent samples $(a_i, b_i) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where every sample is distributed according to either (1) $A_{s,\chi}$ for a uniformly random $s \in \mathbb{Z}_q^n$ (fixed for all samples), or (2) the uniform distribution, which distinguish which is the case (with non-negligible advantage).*

Note that, without errors, both problems are easy to solve using Gaussian elimination. Errors smudge the lattice points and somewhat erase the structure of the lattice, making problems challenging to solve. After Regev's work, most of the lattice-based cryptosystems are based on LWE. Below is the encryption algorithm proposed by Regev.

- **Key Generation:** Sample $A \in \mathbb{Z}_q^{m \times n}$ and $s \in \mathbb{Z}_q^n$, chosen uniformly at random. Sample $e \in \chi^m$ from Gaussian distribution. Compute $b = As + e$. The public-key is $(A, b) \in \mathbb{Z}_q^{(n+1) \times m}$ and the secret key is $s \in \mathbb{Z}_q^n$.
- **Encryption:** For given message M, let $t \leftarrow \{0,1\}^m$. Compute $C_1 = t^T A$ and $C_2 = t^T b + \lfloor \frac{q}{2} \rceil \cdot M$. The ciphertexts are (C_1, C_2).
- **Decryption:** Compute $d = C_2 - C_1 \cdot s$. If the result is close to 0, output 0, else output 1.

As error vectors play an important role in the security of the LWE-based cryptosystem, the importance of implementing the Gaussian sampler has been raised.

2.2. Discrete Gaussian Sampling

Before describing the Gaussian sampling algorithms, the discrete Gaussian distribution on a lattice is defined below.

Definition 7. *(Discrete Gaussian Distribution) Let $L \subset \mathbb{Z}^n, c \in \mathbb{R}^n, \sigma \in \mathbb{R}^+$. Define:*

$$\rho_{\sigma,c}(x) = exp\left(-\pi \frac{||x-c||^2}{\sigma^2}\right) \text{ and } \rho_{\sigma,c}(L) = \sum_{x \in L} \rho_{\sigma,c}(x).$$

The discrete Gaussian distribution over lattice L with center c and parameter σ is defined as

$$\forall x \in L, D_{L,\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(L)}.$$

For $c = 0$, denote $\rho_{\sigma,0}$ as ρ_σ and $D_{L,\sigma,0}$ as $D_{L,\sigma}$.

In lattice-based cryptography, one has to sample a vector from a discrete Gaussian distribution on a lattice $L \subset \mathbb{Z}^n$. This reduces to sampling n number of integers from a discrete Gaussian distribution on $\mathbb{Z}$. There are several methods for sampling numbers from Gaussian distributions. Gaussian sampling algorithms aim to produce a distribution of random numbers that are statistically close to the ideal Gaussian distribution within a certain bound, depending on the security parameter. The distance between the two distributions can be measured by statistical distance. The statistical distance between the distribution generated by Gaussian sampler X and ideal Gaussian distribution Y is defined as follows:

$$\Delta(X, Y) = \frac{1}{2} \sum_{x \in L} |Pr(X = x) - Pr(Y = x)|.$$

Given the security parameter λ, we aim to produce random numbers from X with certain precision so that its statistical distance between ideal Gaussian distribution Y is $\Delta(X, Y) < 2^\lambda$.

Recently, Bai et al. proposed that Rényi divergence can be used as an alternative to the statistical distance in security proofs for lattice-based cryptography [8]. The definition of Rényi divergence is as follows.

Definition 8. *(Rényi divergence of order a) For any two discrete probability distribution P and Q, where $Supp(P) \subseteq Supp(Q)$, and $a \in (0, \infty)$, Rényi divergence of order a is defined as*

$$R_a(P\|Q) = \left(\sum_{x \in Supp(P)} \frac{P(x)^a}{Q(x)^{a-1}} \right)^{\frac{1}{a-1}}.$$

Bai et al. showed that using Rényi divergence in place of statistical distance can result in less precision in the implementation of the security analysis for lattice-based cryptography [8]. In some cases, using Rényi divergence leads to security proofs allowing for taking smaller parameters in the cryptographic schemes. For more information, please refer to [8]. The work of Bai et al. is evidence that Gaussian sampling is an integral part of lattice-based cryptography that affects both security and efficiency. In [13], Micciancio and Walter also stated that the discrete Gaussian sampling can be the main hurdle in implementation, and a bottleneck to achieve good performance. For example, one parameter set of FrodoKEM-640 requires $640 \times 8 = 5,120$ entries in the error matrix, which is equal to the number of samplings needed [18]. Therefore, one has to consider the characteristics of an algorithm and a minimum number of bits required to satisfy the desired security level for efficient implementation. Before presenting the Gaussian sampling methods used in this field, the parameters needed in Gaussian sampling are as follows [19].

- **The standard deviation:** The standard deviation σ determines the shape of the Gaussian distribution. The larger the standard deviation, the more difficult the implementation. That is, the required bits and table sizes increase when the standard deviation is large.
- **The precision parameter:** The precision parameter λ determines how close the Gaussian sampler is to the ideal Gaussian distribution.
- **The tailcut parameter:** Given a target security of λ bits, the target distance from the ideal Gaussian distribution should not be no less than 2^{λ}. Hence, there is no need to sample from $x \in (-\infty, \infty)$. Instead, x is selected from $x \in (-\tau\sigma, \tau\sigma)$ for some tailcut parameter τ, depending on λ.

Cumulative Distribution Table Sampling

There are many sampling algorithms such as rejection sampling and Knuth–Yao sampling [20,21]. However, since the Gaussian sampler plays a crucial role in the performance of LWE-based cryptosystems, CDT sampling is one of the widely chosen sampling algorithms due to its efficiency. CDT sampling is an instantiation of inversion sampling that requires a precomputed cumulative density function table.

The cumulative distribution function (CDF) of a random variable X is defined as follows.

Definition 9. *(Cumulative Distribution Function (CDF)) The CDF F_X of a random variable X is the function given by*

$$F_X = P(X \leq x).$$

The right-hand side of the equation represents the probability of the random variable X takes on a value less than or equal to x. The CDT sampler first constructs the CDT table evaluated at some positive integer points including zero. Given a random number x, sampled from uniform distribution, the idea of the CDT sampler is to return the smallest index i of the precomputed CDT table Ψ such that $\Psi[i] < x \leq \Psi[i+1]$. The length of the table depends on the Gaussian parameters, tailcut, and precision. An example of sampling by CDT is as follows:

Suppose we want to sample from some distribution f, by sampling eight bits uniformly. The first seven bits correspond to a uniform random integer in $[0, 128)$, and the last eighth bit is used to determine the sign of the sampled value. Suppose that, by sampling $0, \pm 1, \pm 2$, and ± 3, we can obtain the desired statistical distance between f and our CDT sampler. Then, we first obtain the probability of $0, 1, 2$ and 3: P[i] = {0.4, 0.3, 0.2, 0.1} according to f. Then, we map this probability to integers between $[0, 128)$: $\boldsymbol{P}[i] = \{51, 38, 26, 13\}$. From this probability density table, we obtain: $\Psi[i] = \{26, 64, 90, 103\}$. Note that we halved the probability of 0 since $-0 = +0$ so that 0 is sampled twice. By sampling $x \in [0, 128)$, CDT sampler returns the smallest $i \in [0, 3]$ such that $x \leq \Psi[i]$. The previous implementation used a binary search to find i efficiently.

Note that the greater the number of bits used to sample, the more accurate the sampler is to the Gaussian distribution. To conclude the section, CDT sampling is a popular choice for implementing lattice-based cryptographic schemes due to their efficiency. Rejection sampling requires high-precision floating-point arithmetic [6]. Knuth–Yao sampling is fast, but it requires large precomputed tables when the standard deviation is large. Therefore, the CDT sampler is widely chosen when implementing lattice-based cryptography.

2.3. Timing Attacks on CDT Sampler

As stated repetitively, the Gaussian sampler is the core element in implementing lattice-based cryptography. If an attacker knows every sampled value from the Gaussian sampler, then the secret key is revealed by simple linear algebra. Hence, resistance against a potential side-channel attack is necessary. In this section, we mainly focus on timing attack performed in the CDT sampler and its countermeasures.

The timing attack proposed by Kocher et al. is a side-channel attack performed by measuring the time of operation of a cryptographic device [22]. The attack exploits the data-dependent characteristic of an implementation that leads to the revelation of the secret key. For example, if an implementation terminates or executes an algorithm depending on the input data, this may be vulnerable to the timing attack. A timing attack on a CDT sampler is performed as follows. Recall that steps 4 to 6 in Algorithm 1 are repeated until it finds k such that $\Psi[k]$ is smaller than the selected random S. Let n be the length of a CDT table. By measuring its time, an attacker can classify the timing of an algorithm to n different groups, where the shortest sampling time will most likely have output zero. This means that naive implementation allows an attacker to determine the sampled value which can lead to the secret key. Hence, Poppelmann et al. proposed a constant-time CDT sampler which guarantees that no information is leaked through timing analysis [7]. Below is the pseudocode of the constant-time CDT sampler implemented in cryptosystems such as FrodoKEM [18].

Algorithm 1 CDT sampling

Require: CDT table Ψ, σ, τ
Ensure: Sampled value S
1: $S \leftarrow [0, \tau\sigma)$
2: $sign \leftarrow [0,1] \cap \mathbb{Z}$ uniformly at random
3: $k \leftarrow 0$
4: while $(S > \Psi[k])$
5: $\quad k++$
6: end while
7: $S \leftarrow ((-sign) \wedge k) + sign$
8: return S

Note that, instead of terminating the algorithm when the index is found, Algorithm 2 searches through the whole table. Algorithm 2 adds 0 to a variable S when the selected random value is less than the table value and adds 1 when the selected random value is greater than the table value. Therefore, the returned value can be equal to the index i such that $rnd < \Psi[i+1]$ even though the whole table has been searched. The timing attack can be eliminated since the execution time of the algorithm is independent from the selected random value.

Algorithm 2 Constant-time CDT sampling

Require: CDF table Ψ of length ℓ, σ, τ
Ensure: Sampled value S
1: $S \leftarrow 0$
2: $rnd \leftarrow [0, \tau\sigma) \cap \mathbb{Z}$ uniformly at random
3: $sign \leftarrow [0,1] \cap \mathbb{Z}$ uniformly at random
4: For $i = 0$ up to $\ell - 1$ do
5: $\quad S$ += $(\Psi[i] - rnd) \gg 15$
6: $S \leftarrow ((-sign) \wedge S) + sign$
7: return S

3. Single Trace Analysis on a CDT Sampler

STA is a statistical power analysis attack using a single power consumption trace. The STA performs by statistically analyzing the distribution of an operation related to a secret key. For example, Amiel et al. recovered the secret exponent in RSA by exploiting the average Hamming weight difference between squaring and multiplication operations [23]. Since STA uses a single power consumption trace, blinded exponent or randomization is meaningless. More examples of these kinds of power analysis attacks include Horizontal Correlation Analysis (HCA) [24], Horizontal Collision Correlation Analysis (HCCA) [25], Recovery of Secret Exponent by Triangular Trace Analysis (ROSETTA) [26], and Hanley et al.'s attack [27]. In this section, we propose that the constant-time CDT sampler is not secure against STA. We theoretically and experimentally analyzed that an attacker can reveal every

value of an error by exploiting a single power consumption trace. This indicates that lattice-based cryptosystems that use a constant-time CDT sampler are vulnerable since an attacker can reveal the secret key by using the recovered error vectors from STA.

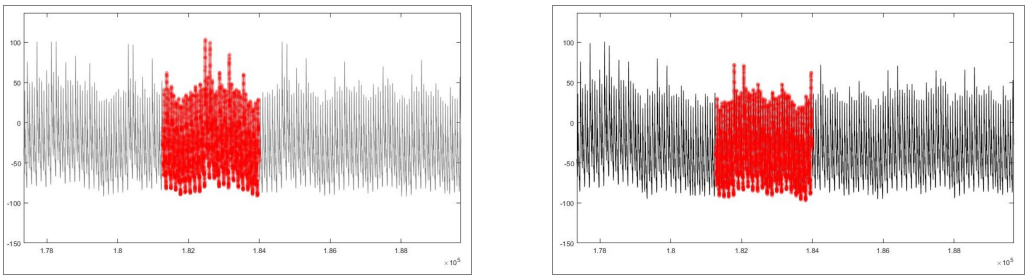
This section is organized as follows: first, we justify our attack by examining raw traces of a CDT sampler. Then, we show that our attack works on the simulated CDT sampler. Finally, we show how to recover the secret key given the sampled values. We demonstrated that any other lattice-based cryptosystems that use the constant-time CDT sampler as in Algorithm 2 is vulnerable to STA.

3.1. Examining Raw Traces

The targets of attack are steps 4 to 6 in Algorithm 2. Due to the application of timing attack countermeasures, step 5 in Algorithm 2 is repeated *len* times regardless of the selected random value. However, note that 1 is added to S when the selected random is larger than the table value. When the selected random is larger than the table value, the result of a subtraction is negative so that the Hamming weight of the value increases. More specifically, note that the significant bits are 1 s when the negative numbers are expressed in two's complements. Therefore, the Hamming weight of the subtraction result is at least 16 bits minus the number of bits used for sampling. On the other hand, when the result of a subtraction is positive, then 0 is added to the value S. Consequently, its Hamming weight is at most the number of bits used for sampling. For a numerical example, suppose the values are expressed in 16-bit integers, and nine bits are used to sample as in FrodoKEM [18]. When the result of a subtraction is negative, the most significant seven bits are 1, while half of the remaining bits are 1 on average. Therefore, negative values will have Hamming weight of 11.5 on average. In a similar manner, when the result of a subtraction is positive, the most significant seven bits are zero, while half of the remaining bits are 1 on average. Hence, positive numbers will have Hamming weight of 4.5 on average.

Due to the gap of Hamming weight between positive and negative number, the sign of the subtracted value (i.e., $\Psi[i] - rnd$) is visible in the power consumption trace. Furthermore, this sign is related to adding 1 or 0 to the S. As a result, instantaneous power consumption increases when 1 is added to the value compared to when 0 is added. We claim that adding 1 and 0 in step 5 can be detected through STA, which allows an attacker to find out which number was sampled by counting this information.

In order to support our claim, we sampled using a CDT sampler that does not cast the subtracted value to an unsigned type. We implemented the CDT sampler in ATmega128 (Atmel, San Jose, CA, USA) and the measurement devices used to obtain the power consumption traces involved a LeCroy HDO6104A oscilloscope (Testforce, Pickering, ON, Canada) sampling at 1 GS/s. The red line in Figure 1a illustrates the power consumption trace when the subtraction result is negative, and the red line in Figure 1b illustrates the power consumption trace when the subtraction result is positive. As depicted in Figure 1, the result was expected as described above. Due to the large Hamming weight of the negative value, an attacker can deduce whether the subtracted result is positive or negative.



(**a**) (**b**)

Figure 1. (**a**) power consumption trace when subtraction result is negative; (**b**) power consumption trace when subtraction result is positive.

3.2. Attack on the CDT Sampler

This section describes how to recover the sampled values through a single power consumption trace by using the result from the previous section. Although Algorithm 2 casts the value to an unsigned integer, the very moment when the difference between the sampled value and the table value is calculated can be captured in power consumption trace. In other words, high power consumption indicates that the negative value is calculated and hence 1 is added to the variable S. On the contrary, low power consumption indicates that the positive value is calculated so that 0 is added to the variable. For the experiment, a CDT sampler in Algorithm 2 with the table of length 9 was implemented in ATmega128. We sampled 100 times by selecting 100 different random value. In order to clearly capture when the CDT is processed, Algorithm 2 was repeated 10 times with the selected random number as input. The power consumption traces were collected using a LeCroy HDO6104A oscilloscope at a sampling rate of 250 M/s. We used a low pass filter on the collected power consumption traces to reduce noise. Figure 2 illustrates the entire power consumption trace of our implementation, which is then zoomed into the region of interest.

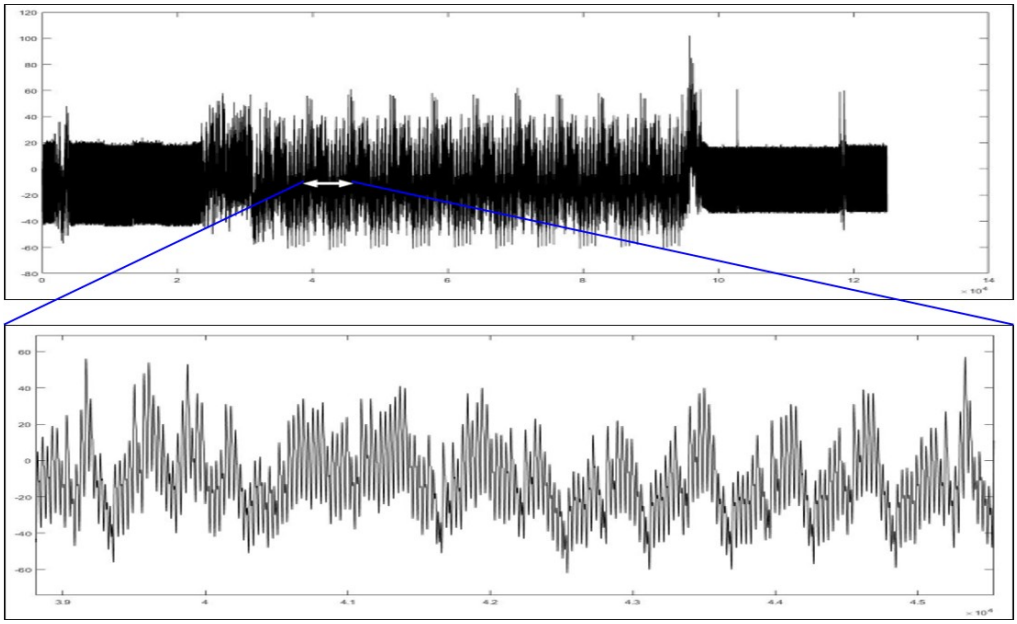
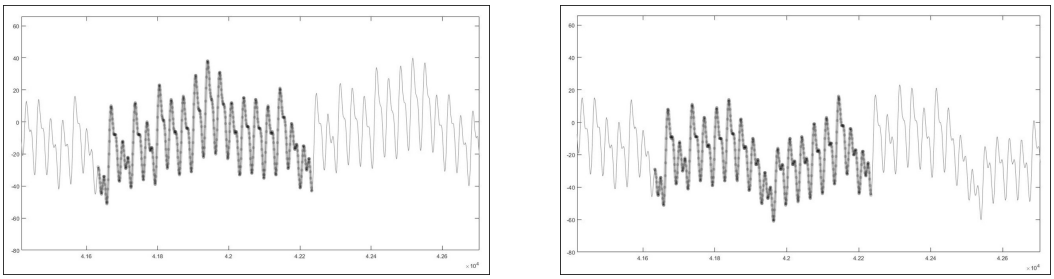


Figure 2. Part of a single execution of the algorithm.

As depicted in Figure 2, we can see 10 identical iterations where its unit is marked with a white arrow line. To better represent the power consumption trace of one execution of the CDT sampler, we cut the power consumption trace into each execution of an algorithm. The bottom of Figure 2 illustrates the power consumption trace when a single value is sampled.

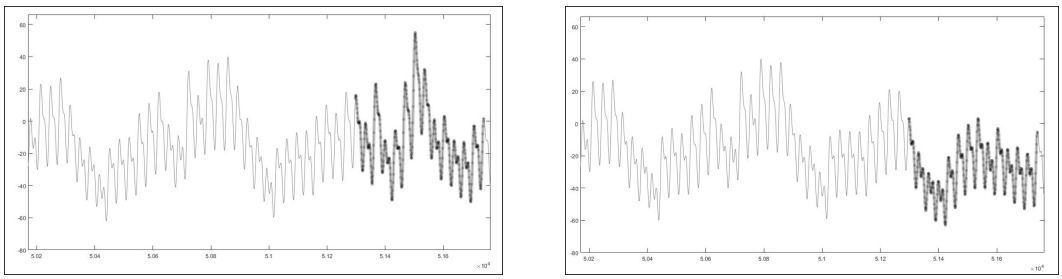
If the power consumption of sampling is identical to every random value, that is, if the algorithm resists against STA, then the power consumption trace of sampling with certain random values will be identical to the power consumption trace of sampling with any other random values. However, this is not the case for the currently used constant-time CDT sampler. Due to the gap of the Hamming weight of the negative and positive values, the sign of the subtracted result is reflected in the power consumption trace. Figure 3a,b illustrates when the subtracted result is negative and positive, respectively. The power consumption trace when 1 or 0 was added is marked as a black line over gray-colored trace as shown in the figure below. As illustrated in Figure 3, the sign of the subtracted result can be detected through power consumption trace.



(**a**) (**b**)

Figure 3. (**a**) power consumption trace when the subtracted result is negative; (**b**) power consumption trace when the subtracted result is positive.

The sign of the sampled bit can also be revealed through STA. Step 6 in Algorithm 2 depends on the sign bit selected in step 3. If 0 is selected, then the result of the sampled value is positive, and negative otherwise. Step 6 in Algorithm 2 flips the variable S when the sign bit is 1 and remains the same when the sign bit is 0. Therefore, this process is visible through power consumption trace. Figure 4 illustrates the power consumption trace when the sign bit is 1 and 0.



(**a**) (**b**)

Figure 4. (**a**) power consumption trace when the sign bit is 1; (**b**) power consumption trace when the sign bit is 0.

For a clear comparison, Figure 5 is the overlapped power consumption trace of Figure 4a,b. The black power consumption trace illustrates when the sign bit is 1 and the red power consumption trace illustrates when the sign bit is 0.

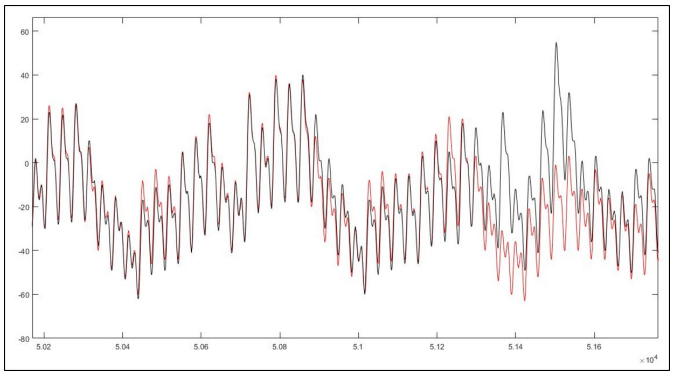


Figure 5. Overlapped power consumption trace of Figure 4a,b. The black line illustrates when the sign bit is 1 and the red line illustrates when the sign bit is 0.

Combining the information in Figures 3 and 4, an attacker can determine the sampled values exactly through a single power consumption trace. To justify our observation, we constructed a tool to

extract the sampled value automatically given a power consumption trace. First, we divided the power consumption trace into one execution of the algorithm. In our experiment, each execution consists of 6160 points. For each partitioned trace, eight iterations were visible, with each iteration consisting of 577 points on average. For example, step 4 starts approximately at 1,154, 1731, 2307, 2883, 3460, 4036, 4615, and 5192.

Figure 6 illustrates the part of the single execution of the algorithm, where the red lines indicate when step 4 starts. In order to visualize the difference of the graph when the subtraction result is positive or negative, power consumption traces when 6 and 2 are sampled through Algorithm 2 are overlapped in Figure 6. The blue power consumption trace illustrates when 6 is sampled and black power consumption trace illustrates when 2 is sampled. In our implementation, the sampled value is equal to the number of the subtractions that result in a negative value out of eight subtractions. Therefore, compared to the case when two are sampled, four more of the subtracted result is negative when six is sampled and is visualized as shown in the blue shaded rectangle as in Figure 6. Particularly, the difference between the power trace when the subtracted result is positive and negative was most noticeable after 157 points for each iteration on average. An example of such difference is illustrated in the blue shaded rectangle in Figure 6, which consists of 284 points on average. This means that the difference between the average power consumption when 0 or 1 is distinct so that an attacker can set a threshold to divide the gap. From our experiment, when the average is greater than –1, we assumed that the subtracted result was negative so that 1 was added to the variable S.

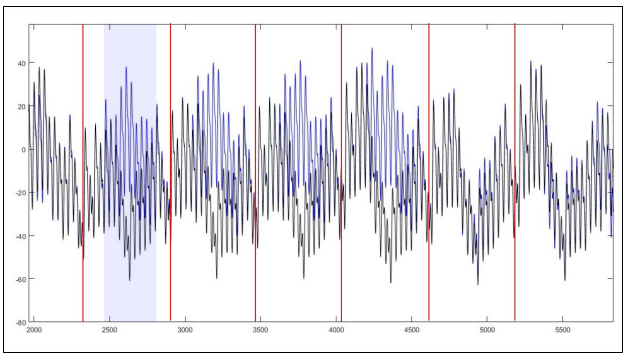


Figure 6. Part of a single execution of the algorithm. The black line is the power consumption trace when 6 is sampled, and red line is the power consumption trace when 2 is sampled. The blue area indicates the difference between the power consumption traces.

Let $x(i)$ be the point where the i-th loop starts. By calculating the average between $x(i) + 157$ and $x(i) + 441$ and comparing it with –1, we were able to deduce whether 1 or 0 was added to the variable. In this way, we were able to recover 1000 sampled values out of 1000 number of power consumption traces in Figure 1 and 100 sampled values out of 100 number of power consumption traces in Figure 2. Since the difference between the average power consumption of negative and positive is apparent, we believe that full recovery of the sampled value is possible even with a large number of samples.

To summarize the attack, STA on a CDT sampler can be performed as follows: suppose that an attacker obtains a power consumption trace that samples 2000 values. The attacker first divides the power consumption trace into each execution of the algorithm, which results in 2000 number of sub-traces. For each sub-trace, an attacker identifies loops of an algorithm. The number of loops must equal the number of table length. By overlapping the sub-traces, an attacker approximates the location where step 5 of the Algorithm 2 is processed. Next, by averaging the power consumption where one or zero adds, an attacker sets a threshold t. For each loop of the sub-trace, an attacker averages the power consumption trace and compares it with the threshold. If the average is greater than t, then the subtracted result is negative so that one is added to S. If the average is smaller than t, then the subtracted result is positive so that zero is added. An attacker can recover the sampled value

since the summation of the number of 1s added is equal to the value sampled. The sign of the sampled value can be revealed with the similar process. Note that the CDT sampler is more vulnerable to such an attack if casting to an unsigned type is not used in step 5 of Algorithm 2. If the right shift of the difference is not unsigned, then –1 is added to the variable S making the gap wider. This results in more visibility in the power consumption graph.

3.3. Remarks on the Proposed Attack

As denoted in Section 2, for the public-key (A, b), a standard LWE-based cryptosystem takes s as the secret key where $b = As + e$ for an error matrix e sampled from the discrete Gaussian distribution. If e is revealed, then $As = b - e$ so that s can be solved in polynomial time using the Gaussian elimination. As we demonstrated above, any lattice-based cryptosystem that uses the constant time CDT sampler is vulnerable to our attack. This includes the FrodoKEM proposed by Bos et al. and Lizard proposed by Cheon et al. [15,18]. The secret key of FrodoKEM is $(s||pk, S)$, where pk is the public-key, s is a random seed chosen uniformly at random, and S is a sample through the Gaussian sampler [18]. Although we can directly obtain S through our STA, the secret key cannot fully be recovered since there is no information relating to s. However, our attack reduces the substantial amount of entropy of the secret key. For a numerical example, consider FrodoKEM-640, which aims at the brute-force security of AES-128. The size of S is equal to $640 \times 8 \times 2 = 10240$ bytes, while the size of s equals to 16 bytes. Considering the public-key, we can recover 99% of the secret key through our attack. On the other hand, the secret key of Lizard is S, where S satisfies the equation $B = -AS + E$ for the public-key (A, B) and E sampled from the Gaussian distribution [15]. Therefore, by performing our STA to recover E, we were able to obtain S through Gaussian elimination.

4. Proposed Countermeasure

Due to the gap of the Hamming weight of negative and positive value, constant-time CDT sampler is vulnerable to STA. In Algorithm 2, when the selected random value is subtracted from the table value, 1 or 0 is added to the variable S if the result was negative or positive, respectively. The negative values will have higher power consumption than the positive values since negative values have larger Hamming weight. Therefore, an attacker can deduce whether 1 or 0 was added by analyzing the power consumption trace. By counting how many 1s were added to the S and recognizing the additional peak after the loop to determine the sign, an attacker can determine the exact sampled value. In this section, we propose an algorithm that resists against STA described in Section 3. Note that we also need to consider the efficiency of the countermeasure, given the fact that numerous samplings are conducted for one execution of the cryptosystem. Our countermeasure is efficient since it is faster than the unprotected algorithm with the use of ROM.

4.1. Table-Based CDT Sampler

Note that the output of the CDT sampler is determined from the moment the random value is selected from the uniform distribution. Let $\Psi[i] = \{26, 64, 90, 103\}$ be the corresponding CDT as in Section 2. The CDT sampler outputs 0 when the random value selected is between [0, 26) and outputs 1 when the random value selected is between [26, 64) and so on. Thus, we can make the look-up table as follows:

From the above table, we can construct the CDF table as $\Phi[i] = \{0, \ldots, 0, 1, \ldots 1, 2, \ldots, 4\}$. A similar method was implemented in [28], where the look-up table was constructed from the start and end addresses of the linear feedback shift register. Sampling values from this reconstructed CDF table are as denoted in Algorithm 3.

Unlike the constant-time CDT sampler described in Algorithm 2, the execution of Algorithm 3 does not depend on the sampled random value. In Algorithm 2, whether the selected random value is greater than the CDT table value was visible in the power consumption trace due to the difference in Hamming weight of positive and negative values. This vulnerability is eliminated in our

countermeasure by precalculating what output of the sampler will be given the random input value. As soon as a random value is selected (step 2 of Algorithm 3) Algorithm 3 uses the random value as the index of the table Φ and outputs the corresponding table value. Since there is no Hamming weight difference or the difference in the execution of an algorithm depending on the selected random value, an attacker cannot extract information from a single power consumption trace.

Algorithm 3 STA-resistant CDT sampling

Require: CDF look-up table Φ, σ, τ
Ensure: Sampled value S
1: $sign \leftarrow [0, 1] \cap \mathbb{Z}$ uniformly at random
2: $rnd \leftarrow [0, \tau\sigma) \cap \mathbb{Z}$ uniformly at random
3: return $S \leftarrow (-sign) \wedge \Phi[rnd] + sign$

However, Algorithm 3 exposes the sign bit through STA in the same manner as in Algorithm 2. In order to prevent the exposure of the signed bit, one can additionally sample one more bit from the uniform distribution and double the length of the table instead of sampling the sign bit separately. For example, suppose that the original algorithm selects eight bits uniformly at random, where the first seven bits correspond to a uniform random integer in [0, 128) and the last bit is used to determine the sign of the sampled value. To use the look-up table for sampling, first, sample 8 bits, and map to integer [0, 128) when the most significant bit is 0 and to $(-128, 0]$ when the least significant bit is 1. Then, Table 1 changes to Table 2 with the length doubled.

Table 1. Sampled values given random input value.

x	**0**	**1**	...	**25**	**26**	...	**63**	**64**	...	**127**
Output	0	0	...	0	1	...	1	2	...	4

Table 2. Sampled values given random input value including the sign.

x	**0**	**1**	...	**25**	**26**	...	**127**	**128**	...	**154**	...	**255**
Output	0	0	...	0	1	...	4	0	...	−1	...	−4

One disadvantage of this algorithm is that it needs to store a look-up table. However, since the output of the Gaussian sampler does not exceed one byte, for λ-bit of precision, we need a table size of 2^λ bytes for Table 1 and $2^{\lambda+1}$ bytes when using Table 2. When applying the look-up table method to lattice-based cryptosystem, table size is approximately 128–1024 bytes for Table 1 and 256–2048 bytes for Table 2. Specifically, for FrodoKEM-640, since the CDT sampler uses a 7-bit random value, the size of Table 1 is 128 bytes, and the size of Table 2 is 256 bytes. Considering the Knuth–Yao table (1080 bytes for LOTUS-128 that aims the brute-force security of AES-128 [29]) used in the lattice-based cryptosystem, we believe that this storage size is acceptable.

4.2. Implementation

To validate the efficiency of our proposed countermeasure, we implemented our table-based CDT sampler on Lizard. The selected parameters for Lizard we used were `CCA_CATEGORY1_N536`, `CCA_CATEGORY3_N816`, and `CCA_CATEGORY3_N952`, which aim at the brute-force security of AES-128, AES-192, and AES-192, respectively. For specific parameters, please refer to [15]. The test was performed on one core of an Intel Core i7-6700 (Skylake) at 3.40 GHz, running Ubuntu 16.04 LTS. The key generation for each parameter was repeated 10^4 times, and its average is given in Table 3.

The CDT table length of `CCA_CATEGORY1_N536`, `CCA_CATEGORY3_N816`, and `CCA_CATEGORY3_N952` are 9, 5, and 6, respectively. As shown in Table 3, our countermeasure becomes more efficient when the

table length of the constant-time CDT sampler is long. In addition, as in Table 3, our countermeasure does not degrade the performance of the cryptosystem since applying our countermeasure does not make any difference in speed compared to the original implementation. Overall, the proposed countermeasure protects the STA without efficiency loss.

Table 3. Comparison of the running time of key generation stage of the original cryptosystem verses countermeasure-implemented cryptosystem represented in milliseconds.

	Lizard		
	`CCA_CATEGORY1_N536`	`CCA_CATEGORY3_N816`	`CCA_CATEGORY3_N952`
Original	213.687	335.144	361.328
Countermeasure (Table 1)	207.979	333.235	356.383
Countermeasure (Table 2)	206.739	330.509	353.166

5. Conclusions

In this paper, we showed that the existing constant-time CDT sampling method for Gaussian sampler is vulnerable against STA and proposed its countermeasure. Since the knowledge of the sampled value can reveal the secret key by using the Gaussian elimination, security regarding the implementation has been recognized as an important problem. However, as efficiency is as important as security when implementing the Gaussian sampler, only constant-time algorithms for Gaussian samplers have been proposed in order to prevent the timing side-channel attacks. However, we proposed that STA was possible for the constant-time CDT sampling method. The constant-time CDT sampling is vulnerable to STA due to the gap of Hamming weight of negative and positive values. This value can be related to adding one and zero to some variable, and, consequently, the sampled value can be revealed. We also recovered the sign bit of the sampled value in a similar way. Therefore, we were able to recover the exact sampled values each time the algorithm was executed. In order to resist STA, we proposed an alternative algorithm for the CDT-based Gaussian sampler. The idea is to use a look-up table that precomputes the sampled value from a given input. Unlike calculating the differences in the constant-time CDT sampler, the proposed countermeasure uses the random value as the index of the precomputed table and outputs the corresponding table value. Since there is no Hamming weight difference or the difference in the execution of an algorithm depending on the selected random value, an attacker cannot extract information from a single power consumption trace. In addition, we implemented our countermeasure in Lizard and check its performance. The proposed countermeasure does not degrade the performance since it works by directly referencing table values.

Author Contributions: S.K. performed the experiments, analyzed the data, and wrote the paper. S.H. analyzed the data and verified the validity of results.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2018-2015-0-00385) supervised by the IITP (Institute for Information & communications Technology Promotion).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **1999**, *41*, 303–332. [CrossRef]
2. Ajtai, M.; Dwork, C. A public-key cryptosystem with worst-case/average-case equivalence. In Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, El Paso, TX, USA, 4–6 May 1997; pp. 284–293.
3. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM* **2009**, *56*, 34. [CrossRef]

4. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 1–23.
5. Peikert, C. An efficient and parallel Gaussian sampler for lattices. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 80–97.
6. Galbraith, S.D.; Dwarakanath, N.C. Efficient sampling from discrete Gaussians for lattice-based cryptography on a constrained device. *Preprint* **2012**.
7. Pöppelmann, T.; Güneysu, T. Towards practical lattice-based public-key encryption on reconfigurable hardware. In Proceedings of the International Conference on Selected Areas in Cryptography, Burnaby, BC, Canada, 14–16 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 68–85.
8. Bai, S.; Lepoint, T.; Roux-Langlois, A.; Sakzad, A.; Stehlé, D.; Steinfeld, R. Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance. *J. Cryptol.* **2018**, *31*, 610–640. [CrossRef]
9. Kocher, P.; Jaffe, J.; Jun, B. Differential power analysis. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–19 August 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 388–397.
10. Bruinderink, L.G.; Hülsing, A.; Lange, T.; Yarom, Y. Flush, Gauss, and Reload—A cache attack on the BLISS lattice-based signature scheme. In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Santa Barbara, CA, USA, 17–19 August 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 323–345.
11. Espitau, T.; Fouque, P.A.; Gérard, B.; Tibouchi, M. Side-channel attacks on BLISS lattice-based signatures: Exploiting branch tracing against strongswan and electromagnetic emanations in microcontrollers. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1857–1874.
12. Karmakar, A.; Roy, S.S.; Reparaz, O.; Vercauteren, F.; Verbauwhede, I. Constant-time Discrete Gaussian Sampling. *IEEE Trans. Comput.* **2018**. [CrossRef]
13. Micciancio, D.; Walter, M. Gaussian sampling over the integers: Efficient, generic, constant-time. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 20–24 August 2017; Springer: Cham, Switzerland, 2017; pp. 455–485.
14. Roy, S.S.; Reparaz, O.; Vercauteren, F.; Verbauwhede, I. Compact and Side Channel Secure Discrete Gaussian Sampling. *IACR Cryptol. ePrint Arch.* **2014**, *2014*, 591.
15. Cheon, J.H.; Kim, D.; Lee, J.; Song, Y. Lizard: Cut Off the Tail! A Practical Post-quantum Public-Key Encryption from LWE and LWR. In Proceedings of the International Conference on Security and Cryptography for Networks, Amalfi, Italy, 5–7 September 2018; Springer: Cham, Switzerland, 2018; pp. 160–177.
16. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A ring-based public key cryptosystem. In Proceedings of the International Algorithmic Number Theory Symposium, Portland, OR, USA, 21–25 June 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 267–288.
17. Peikert, C. A decade of lattice cryptography. *Found. Trends® Theor. Comput. Sci.* **2016**, *10*, 283–424. [CrossRef]
18. Bos, J.; Costello, C.; Ducas, L.; Mironov, I.; Naehrig, M.; Nikolaenko, V.; Raghunathan, A.; Stebila, D. Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016; pp. 1006–1018.
19. Du, C.; Bai, G. Towards efficient discrete Gaussian sampling for lattice-based cryptography. In Proceedings of the 2015 25th International Conference on Field Programmable Logic and Applications (FPL), London, UK, 2–4 September 2015; pp. 1–6.
20. Gentry, C.; Peikert, C.; Vaikuntanathan, V. Trapdoors for hard lattices and new cryptographic constructions. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 17–20 May 2008; pp. 197–206.
21. Knuth, D.; Yao, A. The complexity of nonuniform random number generation. In *Algorithms and Complexity: New Directions and Recent Results*; Academic Press: Cambridge, MA, USA, 1976.

22. Kocher, P.C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 104–113.
23. Amiel, F.; Feix, B.; Tunstall, M.; Whelan, C.; Marnane, W.P. Distinguishing multiplications from squaring operations. In Proceedings of the International Workshop on Selected Areas in Cryptography, Sackville, NB, Canada, 14–15 August 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 346–360.
24. Clavier, C.; Feix, B.; Gagnerot, G.; Roussellet, M.; Verneuil, V. Horizontal correlation analysis on exponentiation. In Proceedings of the International Conference on Information and Communications Security, Barcelona, Spain, 15–17 December 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 46–61.
25. Bauer, A.; Jaulmes, E.; Prouff, E.; Reinhard, J.R.; Wild, J. Horizontal collision correlation attack on elliptic curves. *Cryptogr. Commun.* **2015**, *7*, 91–119. [CrossRef]
26. Clavier, C.; Feix, B.; Gagnerot, G.; Giraud, C.; Roussellet, M.; Verneuil, V. ROSETTA for single trace analysis. In Proceedings of the International Conference on Cryptology in India, Kolkata, India, 9–12 December 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 140–155.
27. Hanley, N.; Kim, H.; Tunstall, M. Exploiting collisions in addition chain-based exponentiation algorithms using a single trace. In Proceedings of the Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, 20–24 April 2015; Springer: Cham, Switzerland, 2015; pp. 431–448.
28. Göttert, N.; Feller, T.; Schneider, M.; Buchmann, J.; Huss, S. On the design of hardware building blocks for modern lattice-based encryption schemes. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Leuven, Belgium, 9–12 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 512–529.
29. Le Trieu Phong, T.H.; Aono, Y.; Moriai, S. LOTUS: Algorithm Specifications and Supporting Documentation. Available online: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions (accessed on 17 September 2018)